HITs における Word 文書の採点プログラム2016年度版の開発

松井吉光
谷口正明
長谷部勝也

キーワード：e-Learning、教材、自動採点、Word、Python

要　　約：本稿は独自開発した e-learning システムである HITs の Word 部門における、文書解析・採点プログラム wt2016.py の開発およびその仕様について述べる。このプログラムは2016年度の情報リテラシー・入門、応用で利用している。

1　始めに

本学名古屋校舎では、2006年度から学生の情報リテラシーレベルの向上を目的として情報リテラシー入門・応用を開講している。情報リテラシー入門・応用では、主に本学で独自開発した e-learning システムである、HITs（Highly Interactive Training system）を教材として利用している。これは Web サーバ上で動作する学習･自動採点･記録（e-Learning）システムであり、Typing 及び Microsoft Office の Word と Excel を対象としている。

wt.py は HITs の Word 部門の採点エンジンで word_test.py の意味である。末尾の py で分かる通り python で書かれている。wt.py は xml 形式で保存された Word 文書（.docx ファイル）と、その文書をどの様に採点するかが書かれたコントロールファイルを読み込んで採点結果を出力する。現用の wt.py は第 III 版であり、フロントエンドの wt2016.py と wtmain2016.py、tool2016.py の三つのファイルからなるプログラム群であるが、以下では総称して wt2016.py と表記する。HITs 運用開始以来10年となる今日、その変遷を振り返っておきたい。

初代の wt.py は xml 形式で表現された Word 文書上で動作する簡易言語であった。即ち、幾つかのコマンドを持ちそのコマンドを組み合わせたプログラム（コントロールファイル）の実行を wt.py が支える。具体的にはサーバ上に蓄積された多数の Word 問題の夫々に特化した同数のコントロールファイル（解析プログラム）が用意される。受講生はサーバから Word 問題をダウンロードし問題中に書かれた指示に従って編集しサーバへアップロードする。コントロールファイル即ち解析プログラムはアップロードされた Word 文書を採点する。この時 wt.py は解析プログラムのインタープリターとして動作するのである。

しかしこの方式は直ちに壁に突き当ることになった。wt.py に表現能力が乏しく、多少とも複雑な解析を実現するためには長大なコントロールファイルを準備する必要が生ずるのであった。表現能力を拡大しようとするとwt.py が肥大化しインタープリター（python）を基礎にした別のインタープリター（wt.py）を構築する愚を犯すことになる。

単純な発想の転換がきっかけになって第II 版への移行が実現した。xml ファイルはツリー構造を持ち、ツリーは xpath の集合と等価である。この時 Word 文書を採点するとは採点対象である特定の xpath が xml ファイルの中に存在するか否かの判定に他ならない。第 I 版がプログラムで実行したことを第II 版はデータ（xpath）の存在 / 非存在の問題に置き換えたのである。

第 I 版は python の xml 解析ライブラリーを使って Word の xml ファイルに特化しHITs 特製の xml ライブラリーを作ろうと努力し第 II 版では xm ライブラリー本来の使用に戻ったと言えるかも知れない。

なお、細かいことを言えば単純に xpath の存在を判定して採点を終わりにするのではなく付随的な様々な判定をしなければならない。例えばその xpath が修飾するテキストの照合、種々の属性値が適切な範囲にあるかの判定等だがそれらは前稿で述べた通りである。

第 II 版の初期では xpath 及び関連判定事項を一行に纏めていた。即ちコントロールファイルは常に一行であった。しかし一行に載せられる情報には限界があり、日ならずして一問三行の形式に移行した。それは

1．xpath 及び関連事項
2．照合すべきテキスト
3．コメント

の形式である。

具体例を示すと

```
/w : document[1]/w : body[1]/w : p[13]\\
/w : pPr[1]\\
/w : ind[1][@w : leftChars="2400"]\\
>> 2200 <= w : leftChars <= 2600
〒464 - 8602 //[2]
左インデント
```

のように記述していた。（なお \\ は、紙面の幅に収めるための見かけ上の改行を表し、実際には改行が存在しないものとする。以降も同様。）2 行目の // の前は採点個所と特定するためのテキストであり、// の後は、採点結果で表示させるテキストである。

この形式に移行してもなお第一行が肥大化するなど、コントロールファイルを書くことに困難を覚える様になった。

第 III 版の主要な変更点はコントロールファイルを xml 化したことにある。本稿はこの点を詳しく述べる。同時に wt.py に通奏低音となって付き纏うテキスト照合とテキスト分断の問題を（前稿［6］との多少の重複を承知の上で）論ずる。

2　コントロールファイルの xml 化

xml 化したコントロールファイルの実例を眺めてみよう。

```
<prob prob_num="1">
  <prob_unit>
    <xpath xtype="para" para_num="11">
      . . . /w : ind[1][@w : leftChars="2400"]
    </xpath>
    <check_param>
      2200 &lt;= w : leftChars &lt;= 2600
    </check_param>
    <check_text>
      〒464 - 8602
    </check_text>
    <comment>
```

```
      左インデント
    </comment>
  </prob_unit>
</prob>
```

見る通り最上位エレメントは prob である。その属性 prob_num=”1” はこの問題が HITsWord 部門の第 1 問であることを示す。第二位のエレメント prob_unit に受講生のアップロードした Word 文書をどの様に採点すべきかを記述している。即ち xpath エレメントの属性 xtype=”para” は照合すべきテキストが para 型（後述）であることを示し、para_num=”11” は第11段落が判定対象であることを示す。xpath エレメントの文字列

```
. . . /w : ind[ 1 ][@w : leftChars=”2400”]
```

は存在を判定すべき xpath 式である。但し前半を … と略しているが、この場合は、

```
/w : document[1]/w : body[1]/w : p[11]\\
/w : pPr[1]
```

のようになっている。

この問題は受講生にインデントを設定することを要求していて check_param 要素の文字列

```
2200 &lt;= w : leftChars &lt;= 2600
```

は許されるインデントの範囲が2200以上2600以下であることを示す（XML では、< と > は予約語であるため、それぞれ「<」、「>」と記述する。）。即ち wt2016.py はアップロードされたファイルに上記の xpath が存在することを調べ存在すれば更にその属性 w : leftChars の値を調べて上の範囲に収まるか否かを判定する。コントロールファイルには

```
w : leftChars=”2400”
```

と記述されているが wt2016.py は値2400 を無視する。check_text エレメントの文字列

```
〒464 - 8602
```

はテキスト照合のデータである（旧形式の第二行）。この文字列は xpath が修飾する受講生のファイルの文字列と正確に一致しなければならない。comment エレメントの文字列

```
左インデント
```

は受講生に判定結果を表示する時そのまま出力されてこれが “ 左インデント ” を設定する課題であることを伝える（旧形式の第三行）。

この例に限れば最上位エレメントの直下に第二位のエレメントを置くのは無駄である。しかし第二位のエレメントが複数現われる場合がある。これを次のコントロールファイル（第13問）で説明しよう。

```
<prob prob_num=”13” units=” 2 ”>
  <prob_unit>
    <grouping>or</grouping>
    <xpath xtype=”para” para_num=”19” >
    . . . /w : rPr[1]/w : rFonts[1] \\
    [@w : eastAsia=”MS P ゴシック”]
    </xpath>
    <check_text> 採用内定通知 </check_text>
    <comment> フォントの種類 </comment>
  </prob_unit>
  <prob_unit>
    <grouping>or</grouping>
    <xpath xtype=”para” para_num=”19” >
    . . . /w : rPr[1]/w : rFonts[1] \\
    [@w : ascii=”MS P ゴシック”]
    </xpath>
    <check_text> 採用内定通知 </check_text>
```

```
    <comment>フォントの種類</comment>
  </prob_unit>
</prob>
```

見る通りprobエレメントに属性units="2"が与えられ、直下に二つのprob_unitエレメントが現れる。

この問題では受講生はダウンロードしたファイル中の文字列“採用内定通知”を指示されたフォントに書き換えることが要請されている。ここで二つの属性名を使い分けた理由は、学生の操作よりできあがるWordファイルに違いができる可能性が否定できないため、どちらかのxpathが存在すれば、Wordで見ると正解になっているという経験則に基づく。そのため二つあるprob_unitのどちらかが不正解になる可能性があるので、どれかが正解なら正解とし他方を無視する必要がある。wt2016.pyはgroupingエレメントの保持する文字列orを読み取って期待通りの判定をする。

グループ化は入れ子にできて例えばprobエレメントの直下に三つのprob_unitが存在して（units="3"）上から順に

```
<grouping>or and</grouping>
<grouping>or and</grouping>
<grouping>or non</grouping>
```

の場合、上から順に二問とも正解であれば第三問を判定せずに正解とし二問のどれかは不正解だが第三問が正解なら正解とする。それ以外即ち最初の二問のどれかが不正解で第三問が不正解なら総合的に不正解と判定する。ここで現われるキーワードor and non又ここには現れていないキーワードnot等の詳細はwt2013.pyの解説[6]で述べた通りである。

既に述べた通りコントロールファイルの記述は常に厄介な問題であった。第II版の初期の一問一行形式ではうまく行かないので一問三行形式に移行したがそれでも問題は解決しなかった。要するにコントロールファイルには意外と大量の情報を載せざるを得なくなっていた。我々は2006年度以来Word文書の採点、即ち構造化されたデータの解析を手がけて来たのだが解析手順を記述するコントロールファイルが又構造化されたデータだという認識は薄かった。そのため自然発生的に定まったコントロールファイルの書式にこだわりwt.pyの中にその読み込みルーティンを埋め込んで来た。

これは開発の硬直化を引き起す。例えば一問四行形式に改めようとすれば最初に読み込みルーティンを書き直さねばならない。三行形式と四行形式のコントロールファイルを共存させることはできない。夫々の行の何処に如何なる情報を埋め込むかは厳しく規制されるからコントロールファイルに新しい情報を付加するには思いがけない困難を伴う。コントロールファイルのどんな些細な変更も読み込みルーティンの手直しを含む大騒ぎになるのである。

xml化した場合コントロールファイルの仕様を拡張して例えば新しいエレメントを追加したとする。この時旧いwt.pyは自動的にそのエレメントを無視するので新旧のコントロールファイルは共存可能である。従って手軽に新しい情報を追記できる。例えばxpathの型が何であるかとか解析すべき段落の番号とかはwt2016.pyは静的に即ちコントロールファイルの中から読み取っている。第II版ではこれを動的に即ちwt.pyがxpathを読んで判定していたがそれは不要の動作（cpu timeの無駄遣い）である。纏めて言えばxml化で得たものは開発の柔軟性である。しかしそれなりのコストも伴う。xmlファイルはエディターで編集できる点で原理的に可読ファイルだが実際は殆ど非可読である（改行を持たないテキスト）。これに加えてxmlファイルは同じ書式の際限も

ない繰り返しを常としファイルサイズは大きくなりがちである。これらは人間の作業として xml 形式のコントロールファイルを手で書くことをほぼ不可能にする。第 II 版から第 III 版への移行には蓄積された三行形式をコンバートするプログラムを書いて凌いだが今後の方針は未定である。

3　wt2016.py の基本動作

wt2016.py は以下のようにコマンドラインプログラムとして実行される。

```
\% wt2016.py word.docx control.xml
```

ここで word.docx は評価すべき Word ファイル、control.xml はコントロールファイルである。control.xml を省略した場合は、採点対象である word.docx に埋め込まれた問題番号を読み取りサーバ上に置かれたコントロールファイルを吏う。wt2016.py の基本動作は

1．xpath の存否。word.docx にコントロールファイルの指定する xpath が存在するか。
2．テキストの照合。xpath が修飾するテキストとコントロールファイルの指定する テキストが一致するか。

の二つである。実際の運用ではこの他に細々した点検動作をするがそれらは前稿で述べたとおりで際立った変更はない。

3.1　xpath の点検

HITs サーバの Word 部門の練習問題 2 の第 1 問（以下例題と言う）は実に簡単な一行問題である[1]。

(1)この行を MS 明朝12ポイントにしな さい。

受講生が HITs サーバから例題をダウンロードした時、この行は MS ゴシックで書かれていてフォントサイズは11ポイントである。受講生はフォントを MS 明朝に変更し、フォントサイズを12ポイントに改めて HITs サーバにアップロードしなければならない。サーバは受講生の編集作業が正しいか否かを判定して受講生に表示する（Web ベースの e-learning）。ファイルのアップロードと殆ど同時に判定が表示されるので受講生は効率的な学習ができる。

この例題でサーバはどれだけの判定をする必要があるだろうか。以下に問題作成者に依る模範解答の部分樹を示す。

```
w : p［12］
  w : pPr［1］
    w : ind［1］
      w : left=456
    w : rPr［1］
      w : rFonts［1］
        w : ascii=MS　明朝
        w : eastAsia=MS　明朝
        w : hAnsi=MS 明朝
      w : sz［1］
        w : val=24
      w : szCs［1］
        w : val=24
```

大まかに言えばアップロードされたファイルに上のツリーが存在すれば正解、存在しなければ不正解である[2]。しかし模範解答との完全一致のみを正解とすれば問題が生ずるだろう。例えばインデントを規定する

```
w : p［12］
  w : pPr［1］
    w : ind［1］
      w : left=456
```

の存否をを判定する必要は無い（過剰判定)。従ってフォントサイズを規定している

```
w : p［12］
  w : pPr［1］
    w : rPr［1］
      w : sz［1］
        w : val=24
      w : szCs［1］
        w : val=24
```

及びフォントの種類を規定している

```
w : p［12］
  w : pPr［1］
    w : rPr［1］
    w : rFonts［1］
      w : ascii=MS　明朝
      w : eastAsia=MS　明朝
      w : hAnsi=MS 明朝
```

の二つの存否を判定すれば良い。例えばフォントの種類が確かにw : ascii=MS 明朝であることはアップロードされたファイルにxpath

```
/XA/XB/w : rFonts[1][@w : ascii="MS 明朝"]
```

が存在することを見れば確認できる。但し略記した/XA/XBは

```
XA=w : document[1]/w : body[1]
XB=w : p[12]/w : pPr[1]/w : rPr[1]
```

の意味である。これに併せてフォントサイズが11ポイント即ちw : val=24であることを確認するにはxpath

```
/XA/XB/w : sz[1][@w : val="24"]
```

が存在することを見れば良い。実際の採点では説明した三つのxpathの他に念の為更に四つ、合計六つのxpathの存在を確認し、そのどれかのxpathが存在しないなら不正解と判定している。なお、正確に言うと夫々のxpathを確認する時、同時にそのxpathが修飾する文字列を照合している（次章の主題)。

今の例について少し観点を広げれば六つのxpathを持ち出す理由は何かと言う疑問が生ずる。それは即ちアップロードされたファイルと模範解答ファイルにどの程度の一致があれば正解と判定をするかの問題であり、愛昧な領域が残り残念ながら経験的知識に頼っている部分も多いのである。

3.2　テキストの照合

テキストの照合とはxpathが修飾する文字列が（今の例題で）確かに

（1）この行をMS 明朝12ポイントにしなさい。

であることの確認である。不時な受講生がいてここを

はいはい、MS 明朝12ポイントにしました。

等と書き換えても正解とするのは不都合である。それ故受講生には問題ファイルの文字列の変更を禁じている[3]。受講生がこのルールを破った場合は直ちに不正解と判定する。細かいことを言うとこのルールのボーナスとして段落番号を信頼する判定が可能になる。即ち今の例では第12段落を示すw : p［12］は受講生によるファイルの編集で変化しない。

受講生にテキストの書き換えを禁じてもWordの内部事情によるテキストの分断が起きてこれがテキストの照合を困難にしている。この問題への対処のためHITsはWord文書のテキストを次の三種類に分類して扱っている。

1．段落全体（para 型）
2．段落中の部分文字列（run 型）
3．上記以外（exotic 型）

ここで（1）para 型は xpath が

. . . /w : p/. . .

の形をしており w : p の前後に w : r が含まれない場合であって、テキスト分断は問題にならない。段落に現われる文字列を機械的に連結したものが照合文字列である。

（2） run 型は xpath が

. . . /w : p/w : r/. . .

の形をしており w : p の直下に w : r が現われる。この場合は同じ段落に幾つかの w : r が存在できテキスト分断が問題になる。

（3） exotic 型は xpath に w : p が現れず

. . . /w : r/. . .

の形をしている。この型はフッター、ヘッダー、ページ番号の指定等の特殊な場合に出現する。この場合も para 型と同じで分断は問題にならない。

3.3 テキスト分断

テキスト分断とは何かを例題の模範解答の場合に示す。

```
w : p［12］
  w : r［1］
    w : rPr［1］
      w : rFonts［1］
        w : ascii=MS　明朝
        w : eastAsia=MS　明朝
        w : hAnsi=MS 明朝
        w : hint=eastAsia
      w : sz［1］
        w : val=24
      w : szCs［1］
        w : val=24
      w : t［1］
        "（1）この行を MS 明朝 1 "
  w : r［2］
    w : rPr［1］
      w : rFonts［1］
        w : ascii=MS　明朝
        w : eastAsia=MS　明朝
        w : hAnsi=MS 明朝
      w : sz［1］
        w : val=24
      w : szCs［1］
        w : val=24
    w : t［1］
      "2"
  w : r［3］
    w : rPr［1］
      w : rFonts［1］
        w : ascii=MS　明朝
        w : eastAsia=MS　明朝
        w : hAnsi=MS 明朝
        w : hint=eastAsia
      w : sz［1］ w : val=24
        w : szCs［1］
      w : val=24
    w : t［1］
      " ポイントにしなさい。"
```

ここに見る通り文字列

（1）この行を MS 明朝 12 ポイントにしなさい。

は三つの文字列に分断され、同じ構造の三つのツリーが積み重なっている。

```
w : p［12］
  w : r［1］
    . . .
    w : t［1］
      "(1) この行を MS 明朝 1 "
  w : r[2]
    . . .
    w : t［1］
      "2"
  w : r［3］
    . . .
    w : t［1］
      " ポイントにしなさい。"
```

但しここで. . . は

```
w : rFonts［1］
  w : ascii=Ms　明朝
  w : eastAsia=Ms　明朝
  w : hAnsi=Ms 明朝
  w : hint=eastAsia
w : sz［1］
  w : val=24
w : szCs［1］
  w : val=24
```

の略である。

Word は何故テキスト分断を起こすのだろう。Word 文書の中に文字列 A があって未だ修飾が施されていないとする。当然 Word は A を一つの w : r のもとに格納している。ここで A の前半部分 B に修飾（例えば下線）を施し後半 C には何もしないとする。

$$A=\underline{B}+C$$

この結果、三つの w : r が生成されて夫々 B と C を格納するだろう。最初の w : r が B の修飾（下線）を規定し次の w : r は C に何の修飾も規定しない（もとのまま）。ツリーを示すと

```
w : r［1］
  *** ここで下線が規定される
  w : t［1］
    B
w : r［2］
  . . .
  w : t［1］
    C
```

となる。次に Word を編集操作して B の修飾を取り消したとする。B の修飾 *** は旧に戻り、こうして B,C に同じ部分ツリー. . . が生まれる。

```
w : r［1］
  . . .
  w : t［1］
    B
w : r［2］
  . . .
  w : t［1］
    C
```

この段階で Word は編集作業を終えるだろうか？それともう一歩進んで三つの w : r を一つの w : r に纏める作業、即ち真の undo を実行してファイルを最初に戻すだろうか？

```
w : r［1］
  . . .
  w : t［1］
    A
```

尤もらしいのは前者である。なぜなら Word の仕事は xml ファイルの情報を紙又はモニターに出力することであり xml がどの様な表現を持っていようと構わないからである。こうしてテキスト分断が起きる[4]、そしてこれは HITs にとっては大きな迷惑である。分断を修復しないとテキスト照合ができない！。

3.4 分断の修復

再びテキスト分断が起きた場面に立ちかえる。

```
w : p［12］
  w : r［1］
    . . .
  w : t［1］
    "(1) この行を MS 明朝 1 "
w : r［2］
  . . .
  w : t［1］
     "2"
w : r［3］
  . . .
  w : t［1］
    " ポイントにしなさい。"
```

この例によって分断から回復する作業が分かる。

修復のルール

w:p の直下に連続する幾つかの w:r があって夫々の部分樹. . . が同じなら w:t に格納された文字列を繋ぐ

しかし“同じ”の意味を厳密にとると修復に失敗する。実際、運用では以下のエレメント

w : bookmark
w : spacing
w : commentRange
w : bCs
w : proof
w : iCs
w : smartTag
w : lastRenderedPageBreak

及び属性

w : cs
w : hint
w : rsidR

を無視した上で“同じ”か否かを判断している。この様な例外規則はこれ又経験的知識である

4 まとめ

本稿では、HITs システムの一部で、Micosoft Word ファイルの自動採点プログラム wt.py の2016年度版 wt2016.py の開発およびその仕様について説明した。wt.py は毎年改良を加え、採点の精度の向上に努めるとともに、問題作成の簡単化を模索している。ただ､現時点でも問題作成､特に採点コントロールファイルの作成は Word の XML に精通し、Word の操作による変化についての知識が必要であり、とても誰でも作成できるレベルではない。

今後も引き続き改良を加え、より精度の高い採点が可能なプログラム、より学習効果が期待できるプログラムを目指して開発していくとともに、問題作成の簡単化を進めていきたいと考えている。

注

1 練習問題 2 には全部で 8 個の問題があってこれはその最初の問題である
2 これに加えて関与するテキストの照合を行う。次章参照
3 残念なことに Word には文書の一部を書き換え不能にするオプションがない。
4 テキスト分断が起きる状況は多様でその詳細は不明である

参考文献

[1] Standard ECMA-376 Office Open XML File Formats, http : //www.ecma-international.org/publications/standards/Ecma-376.htm
[2] XML Path Language バージョン 1.0 W3C 勧告、http : //www.w3.org/TR/1999/REC-xpath-19991116
[3] Office 2003 XML リファレンススキーマ、

http：//www.microsoft.com/japan/office/previous/2003/xml/default.mspx
[4] "Excel, Word 自動採点システム HITs の構築と運用" 岩田員典、他、愛知大学情報メディアセンター（20）No.1（2010）
[5] "HITs における Word 文書の採点プログラムの開発" 長谷部勝也、松井吉光、谷口正明 愛知大学一般教育論集（40）25-40（2011）
[6] "HITs における Word 文書の採点プログラム 2013 年度版の開発" 長谷部勝也、松井吉光、谷口正明、愛知大学一般教育論集、(45) 41-53（2013）